Optimization of a Hexapedal Robot for Legged Locomotion

Undergraduate Thesis

Presented in Partial Fulfillment of the Requirements for Graduation with Research Distinction in the Department of Mechanical and Aerospace Engineering at The Ohio State University

By:

Isaac Khan

Thesis Committee

Dr. Manoj Srinivasan, Advisor

Dr. Krishnaswamy Srinivasan

April 2019

Abstract

Humans have a great ability to self-optimize their movements and adapt to various conditions. This is related to human's ability to optimize their motor actions to increase energy efficiency and improve stability and robustness of performance. An example of this can be seen in walking. When humans are walking and arrive at a different surface condition, ice for example, they are able to quickly adapt and optimize an efficient way to move across the ice. This adaptation includes changes in the walking styles, extension of arms for increased balance, as well as a reduce in speed to maintain balance. Current robots lack such quick adaptive ability as many robots are hard coded for particular scenarios in locomotion. In this thesis, our objective is to explore adaptation and optimization in robots in a simplified setting. We use an open source hexapedal robot, miniRHeX. We examined how walking speed performance depended on a few parameters of the leg motion controllers, specifically by sweeping through an acceptable range of each parameter; we considered parameters such as the relative phase between the legs, the PD gains of the proportional and derivative gains, and the walking period. We then implemented a constant step size version of gradient descent optimization and found substantial improvement in walking speed in just 2-3 iterations. The methods show promise, but suggest implementing a line search, or perhaps using coordinate descent to be robust to trial-to-trial variability in measured performance.

Acknowledgements

First and foremost, I would like to thank my parents for all the sacrifices they have made to give me opportunities they never had. As a first-generation American, I witnessed first-hand the struggles they faced to provide for my sisters and I. It is because of my entire family pushing me to do my best and supporting me through all of the challenges I had faced that I am able to achieve any of my goals.

I would like to thank Dr. Manoj Srinivasan, Dr. Krishnaswamy Srinivasan, Dr. Satyanarayana Seetharaman, and Dr. Lisa Abrams. Dr. Manoj Srinivasan for his countless hours of support and guidance to help me find the path when I was lost. He taught me to see robots through an analytic lens that helped me focus my goals. Dr. Krishnaswamy Srinivasan for showing me the power of controls and asking me thought-provoking questions. Dr. Seetharaman for telling me I am capable. I found myself often doubting my abilities, and it was Dr. Seetharaman's guidance that eased those self-doubts. Lastly, I would like to thank Dr. Lisa Abrams for helping me by giving me access to professional development resources that allowed me to learn lessons that I will use for the rest of my life.

I would also like to thank Larry Antal for his expert machining help. Lastly, I would like to thank the Center for Design and Manufacturing Excellence for giving me access to their 3D printers.

Table of Contents

List of Figures

Chapter 1: Introduction

Legged robots and wheeled robots are both used in various applications. Historically, wheeled robots are higher in efficiency and often utilized in projects where flat, fixed terrain is available. [Tucker, 1970]. There are numerous examples of how these robots are created and implemented. The auto industry heavily researches autonomous wheeled vehicles (e.g., cars and trucks), which would qualify as wheeled robots [Greenblatt & Saxena, 2015]. Legged robots have multiple functional advantages and disadvantages when compared to wheeled robots. Legged robots are harder to control and have lower efficiencies. However, wheeled robots are reliant upon relatively flat and non-variable terrain, whereas legged robots thrive in variable terrain [Kimura & Cohen, 2003]. Given that humans move using legs, legged robots (especially two legged) may also be ideal for environments built for humans [MIT Darpa Challenge Team, 2012-2015].

Legged robots have multiple applications in transportation. Assistive technology that could benefit the elderly in mobility is a direct application of this research [Kwa et al, 2009]. Further understanding ways to optimize legged robots with a large number of input parameters will continue to create safer robots that could ultimately be the only mode of mobility for this demographic [Zhang et al, 2017]. In addition, these legged robots can be used for disaster relief in getting food/supplies to people in need when it is too dangerous for people to go [MIT Darpa Challenge Team, 2012-2015].

To develop robotic technology to be more practical, legged robots need to be more energy efficient and optimized for numerous input parameters and work robustly in a variety of situations. In this research project, we will consider optimization of walking and running gaits in a hexapedal robot and develop robot-in-the-loop optimization methods to maximize average velocity. There have

been only a few attempts at optimizing robot energy consumption directly using robot-in-the-loop optimization (to be described below) and there is much that is not understood regarding how best to change robot behavior to maximize outputs such as average velocity or energy efficiency. Bhounsule et al [2014] used a detailed model of a bipedal robot, performed high-accuracy optimization, and implemented the optimal strategies in their robot. Most other attempts at optimization have also relied on detailed mechanics-based modeling of the robot [Prasanth et al, 2016, Waldron et. al, 2008, Wang et. al, 2012, Hemker et. al, 2009]. Here, in contrast, we will attempt to use the movements and measurements of the robot directly in the optimization procedure, alternating between robot movements and numerical optimization procedures. Previous attempts at hardware-in-the-loop optimization of robots looked at only flat ground walking or considered only simple metrics [Weingarten et al, 2004, Hemker et. al, 2006]. Further, the mass-normalized energy cost of those robots were still many times that of animals [Weingarten et al, 2004].

Implementing a self-automated learning algorithm has been a goal and focus for researchers interested in advancing optimization techniques in robots. Weingarten et al [2004] investigated optimization techniques using a Nelder-Mead descent on gait parameters. Their research consisted of using RHex – a hexapedal robot – for automated optimization for parameters such as average velocity and specific resistance. Weingarten et. Al [2004] hand tuned parameters to achieve optimal velocities of 0.8 m/s and specific resistance 2.0. The Nelder-Mead descent was then used to optimize these parameters and the velocity increased to 2.7 m/s and a specific resistance of 0.6. These tests were conducted in a lab environment in which data was collected using on board cameras. This data was then mapped to controllers and run through a state machine used for automated gait optimization. The state machine used for optimization had variant loops in which

optimization would occur. If primary loops were not successful, backup loops would be implemented in which human intervention would be needed. Though these tests were conducted to optimize average velocity and power consumption using specific resistance, surface condition was something that had yet to be further investigated. Further, only one optimization algorithm was explored, namely Nelder-Mead, which is a heuristic technique without a rigorous convergence proof. So, here, we explore similar robot optimization with simpler techniques, specifically "gradient descent" or "gradient ascent" [Nocedal & Wright, 1999] which has a more intuitive basis (hill climbing) and also has a rigorous mathematical foundation. This work also complements and is conceptually simpler than other recent work on robot optimization using various reinforcement learning techniques [Smart & Kaelbling, 2002, Long-Ji Lin, 1991].

Choosing and constructing a legged robot is the first step to perform these experiments in robot optimization. Here, we chose to build an open source robot called MiniRHex, shown in figure 2.1 [Barragan et al, 2018]. The MiniRHex is a hexapod with six curved legs, each with an individual DC motor, figure 4.3. Further technical details of the robot are described in the methodology chapter. With regards to a controller, a mixture of both closed loop and open loop control methods will be implemented for actuation of the motors [Waldron et. Al, 2008]. We will explore different controller structures and actuation profiles so as to determine what actuation profiles and commanded reference trajectories will be optimal for the MiniRHex. Our project not only contributes to the mechanics of this particular legged robot, but also tests the feasibility of simple optimization algorithms for self-adaptation of legged robots.

Chapter 2: Methods

In this chapter, we describe the physical robot used to perform the optimization calculations, the high-level controller structure for the locomotion controller (whose parameters we change), and the optimization procedures considered.

Physical robot platform. To perform these optimization experiments, we needed to construct a legged robot that can be driven with a locomotion controller that had sufficiently many control parameters that could be varied during the optimizations. The open source hexapedal robot MiniRHex [Barragan et al 2018] from a robotics lab at Carnegie Mellon University demonstrated all of the qualities that were needed for this project and was chosen as the robot to complete this project: simple and mechanically robust, comes with a default (if inefficient) controller, which could be improved upon. In addition, [Barragan et al 2018] optimized cost of the MiniRHex for the purpose of creating an open source robot that is more accessible to students because of its cost of under $300 USD. See figure 2.1 for a 3D rendering of the robot.

Robot components were ordered from various vendors in order to minimize costs; the vendors included McCmaster-Carr, Amazon, Ebay, and RobotIS. The motors utilized in this project were Dynamixel XL-320 series motors. It was used in part to its versatility of integration with multiple programs and internal control structure as seen in figure 4.2 in the appendix. There was a choice of motors that could be selected for this project. MiniRHex was designed to fit a Dynamixel XL-320 motor with dimensions 24 mm x 36 mm x 27 mm; however, there was a choice of a larger motor, Dynamixel XL-430 series, that would give MiniRHex a larger stall torque of 1.5 Nm at 61 rpm compared to the 0.39 Nm at 114 rpm. It was decided to continue with the Dynamixel XL-320

because the design was fit to this motor; it was also deemed that the torque necessary to move the robot was sufficient using the Dynamixel Xl-320. The length and width of MiniRHex were 18.6 cm x 10 cm. The height varied slightly due the springiness of the legs but was approximately 7.62 cm when standing.

A large portion of the robot was built using 3-D printed parts: the motor covers, the adapters that extended from the motor to the legs, as well as the legs. In addition, acrylic was used as the base of the robot instead of 3-D printed material.

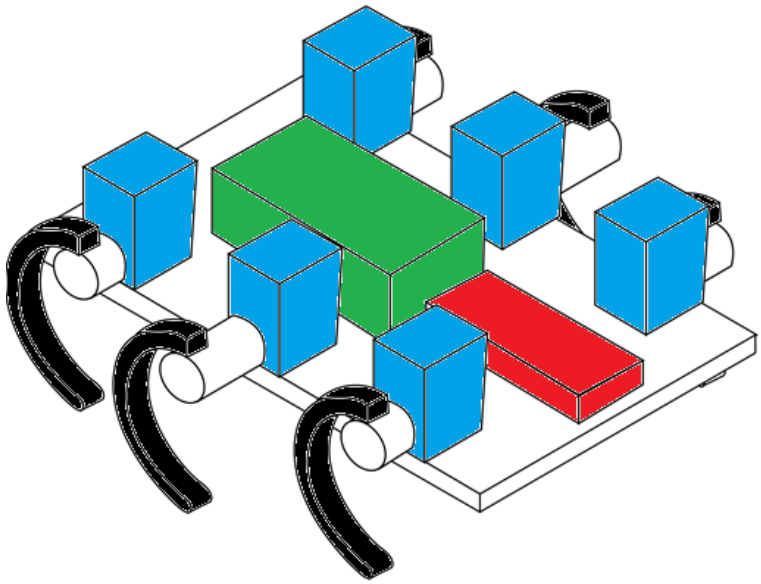

Figure 2.1: Adaptation of MiniRHex used to highlight components. The components are described as the following: black are the 3-D printed legs, blue are the Dynamixel XL-320 motors, green is the battery, and red is the Robotis Open CM microcontroller

A small but time-consuming hurdle was encountered was when a screw holding the adapter extension piece 3 in figure 4.1 was too short. The suggested screw by the open source designs was not standard size. This issue delayed testing the robot as the integrity of the legs held on to the robot was dependent on this screw. After getting expert help, the correct screw was found and ordered.

Locomotion controller for MiniRHex. At its core, the default locomotion controller for the walking gait utilizes a PD controller for leg angular position. Specifically, the controller has a reference or desired angular position as a function of time for each leg. Figure 2.2 illustrates the desired angles. This desired angular position is generally the same for all legs, except for a systematic phase shift between the legs. The PD controller tries to enforce this desired angular motion, but not very aggressively, so that the controller is relatively compliant. The desired angular position is periodic, with a fixed walking period, one of the input parameters to be tested.

Note that the desired angular position (figure 2.2) has two different phases. From t = 0 to t = q_1, a steeper slope is seen as compared to t = q_1 to t = q_2. This is to give the leg more time and reduced velocity when it is in contact with the ground. This reduced velocity when in contact with the ground improves traction of the leg and improves motor torque capabilities. The PD control signal was defined as $\tau = K_P(\theta_{Desired} - \theta_{Actual}) + K_D(\theta_{Desired} - \theta_{Actual})$ (1). The gains for this PD controller were initially set to $K_P = 0.04$ and $K_D = 1$. Only a handful of proportional gains were tested.

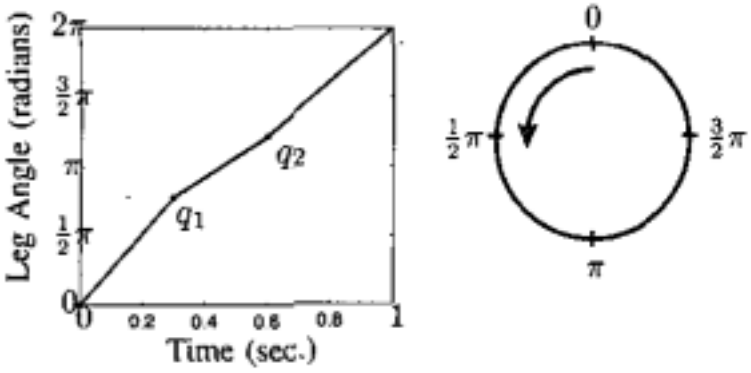

Figure 2.2: Example of walking period used to define what angular position leg should be at with respect to time. Utilized in PD Controller. [Weingarten. et al, 2004].

Performance versus univariate changes in control parameters. We tested how the speed depends on a few different parameters by changing them through a range of values systematically. The parameters examined were leg phase coherence, walking period, and PD gains.

The first tests that were conducted iterated the leg phase coherence. Phase coherence describes the difference in when one leg hits the ground compared to when another leg hits the ground. Similar to how humans walking one leg after the other, the robot was coded to walk with an alternate trip gait, so that legs 1, 3, 5 moved together and legs 2, 4, 6 moved together (as denoted in figure 4.1), but these two sets of three legs moved with a phase difference between each other. A phase difference of 0 or 1 implies all legs moving together, whereas a phase difference of 0.5 implies the two sets of three legs moving at 180 degrees out of phase resulting in a true alternate tripod. We computed the average walking velocity over 3 feet, starting from rest, for phase differences ranging from 0 to 1 in steps of 0.1. At each phase difference, we repeated the walking trial three times, so as to get a better estimate of average performance and also quantify the trial to trial variability.

The velocity was computed by a human measuring the time (using a stopwatch) taken for covering the fixed distance traveled. These tests were performed on two surfaces, once on carpet and once on concrete, both on horizontal ground.

Next, we repeated the experiments to characterize the walking speed as a function of the walking period, all else fixed at their nominal values. This was tested between walking periods of 400 ms and 2000 ms, in steps of 320 ms. Finally, we explored the dependence of the speed on PD gains, although we did not perform a complete sweep of these parameters.

Optimization via gradient descent. We examined the effectiveness of the method of gradient descent or ascent to improve and optimize controller parameters (for fixed controller structure outlined earlier). Gradient descent is a first-order optimization algorithm, which steps iteratively toward a local minimum [Nocedal & Wright, 1999]. We can visualize this as follows: you are at the summit of a large mountain and your goal is to reach the base as quickly as possible. To do this, at every step, you take a step along the steepest direction, thus following the steepest path going down. If you follow this until you reach the base, you will have completed a method of gradient descent.

In general, given a cost function f(x1, x2, x3, …) to be minimized with respect to the variables x1, x2, x3, …, gradient descent proceeds by changing the values of the variables in proportion to the partial derivatives relative to those variables: $\partial f / x_i$. That is,

$$\Delta x_i = -\frac{\partial f}{\partial x_i}\alpha$$

where α is a step size. The negative sign in the above equation ensures that it is a gradient descent algorithm, as opposed to a gradient ascent algorithm; gradient descent uses negative slopes while gradient ascent uses a positive slope. For instance, when we optimize the time duration T for traveling a given distance and optimize relative to the phase difference variable $\emptyset$, we use the following iteration at every step.

$$\Delta\emptyset = -\frac{dT}{d\emptyset}\ \alpha \quad (2)$$

This change then adds to the current best iterate of the phase difference to find a new phase difference value to test at.

$$\emptyset = \emptyset \pm \Delta\emptyset \quad (3)$$

We used a simple variant of gradient descent such that the changes in the control parameters suggested by gradient descent did not exceed 10% of the overall range of the control parameter. This condition was activated only if the step was larger than this upper limit. If this condition was invoked, the step size factor α was then modified to be described in eq. 4.

$$\alpha_{new} = \frac{Upper\ Limit}{\frac{dT}{d\emptyset}} \quad (4)$$

We considered doing line searches using a quadratic approximation along the gradient direction to get a sufficient decrease, but did not implement such a line search. While creating a self-optimizing code with limited to no human input was the initial goal of the project, we instead performed the gradient descent optimization for both 1-D and 2-D parameter sets using hand calculation. Both these gradient descent optimization calculations involved optimizing over one or two phase

difference variables. The overall procedure for such human-intervention driven optimization is provided in figure 2.3 below.

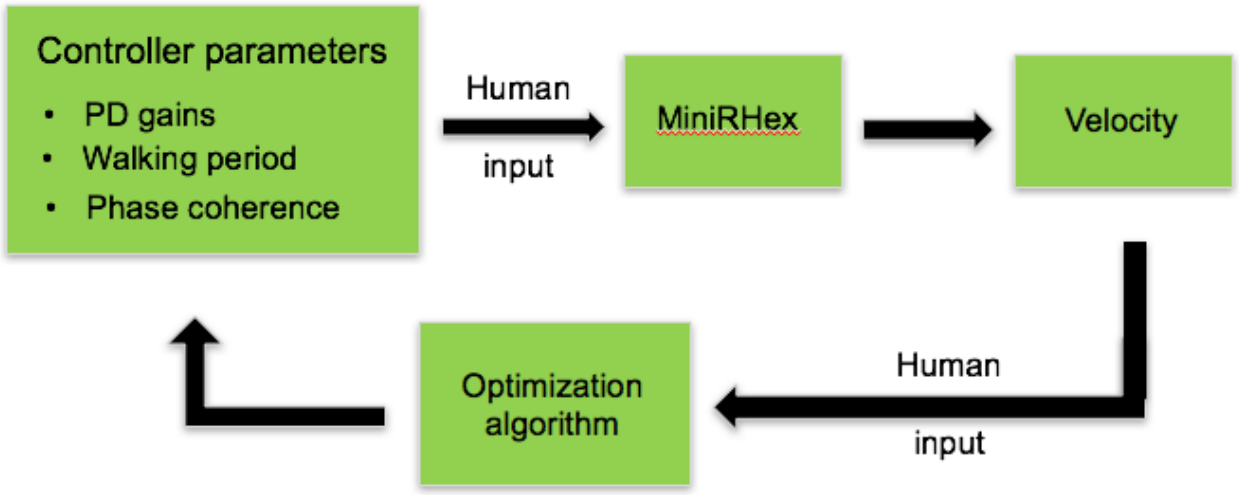

Figure 2.3: Block diagram of optimization flow

Chapter 3: Results

This chapter describes results from measuring performance as a function of various single parameters and from performing limited optimizations using gradient descent.

Univariate parameter sweeps.

Dependence on phase coherence in an alternate tripod gait on carpet. We determined average walking speed as a function of phase coherence of the six legs, with sets of three legs moving together. First, we performed these tests on carpet. Figure 3.1 shows how the speed changes with the phase difference between these two sets of three legs. Based partly on anecdotal remarks in prior studies, it was expected that there would be an optimum at 180 degrees out of phase for this walking gait. This was hypothesized because humans use a near 180 degrees out of phase walking gait of one leg after another, as do six legged animals such as cockroaches. Indeed, as seen in Figure 3.1, there we have maximum speed performance at 0.5, which corresponds to the 180 degrees out of phase that was hypothesized. We also notice some small trial to trial variability in the measured speed, which may be a mixture of human timing errors and true trial to trial variability in the robot's performance.

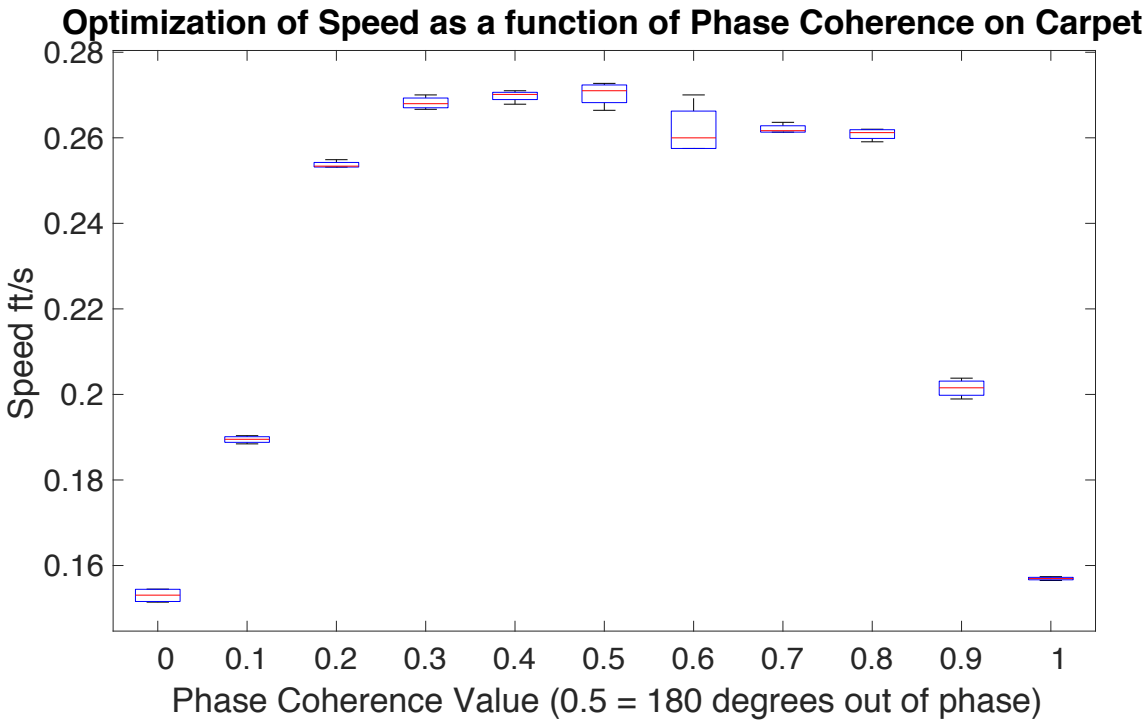

Figure 3.1: Velocity as a function of phase angle difference for a 2 sets of 3 legs walking gait on carpet. This was plotted by the default boxplot in Matlab. The red lines indicate the average, inside the box represents 50% of the data, and then 25% above and below 50%. There are no error bars on phase coherence, the width of the points has no significance.

Dependence on phase coherence in an alternate tripod gait on concrete. An identical test was conducted on concrete, so as to compare how various surface conditions impacts on phase coherence. It was hypothesized that a similar optimal value (180 degrees out of phase) would be obtained. Surprisingly, figure 3.2 shows that the optimum leg phase difference is not equal to 0.5 or 180 degrees, but is systematically different from it. The optimal phase coherence instead was at approximately 252 degrees. We hypothesize that the finiteness of the distance traveled and an imperfect startup that causes the robot to jump forward is the cause of the shift in optimal value. This unusual startup was also noted in the carpet experiments but was fairly insignificant on carpet compared to that on concrete. We see that not only is the optimal phase coherence value is different, but the optimal speed is also significantly higher on carpet, specifically, about 0.35 m/s

on concrete versus 0.27 m/s on carpet. Thus, we see that a given controller can have difference performances based on the walking surface and different controller parameters can be optimal for different surfaces.

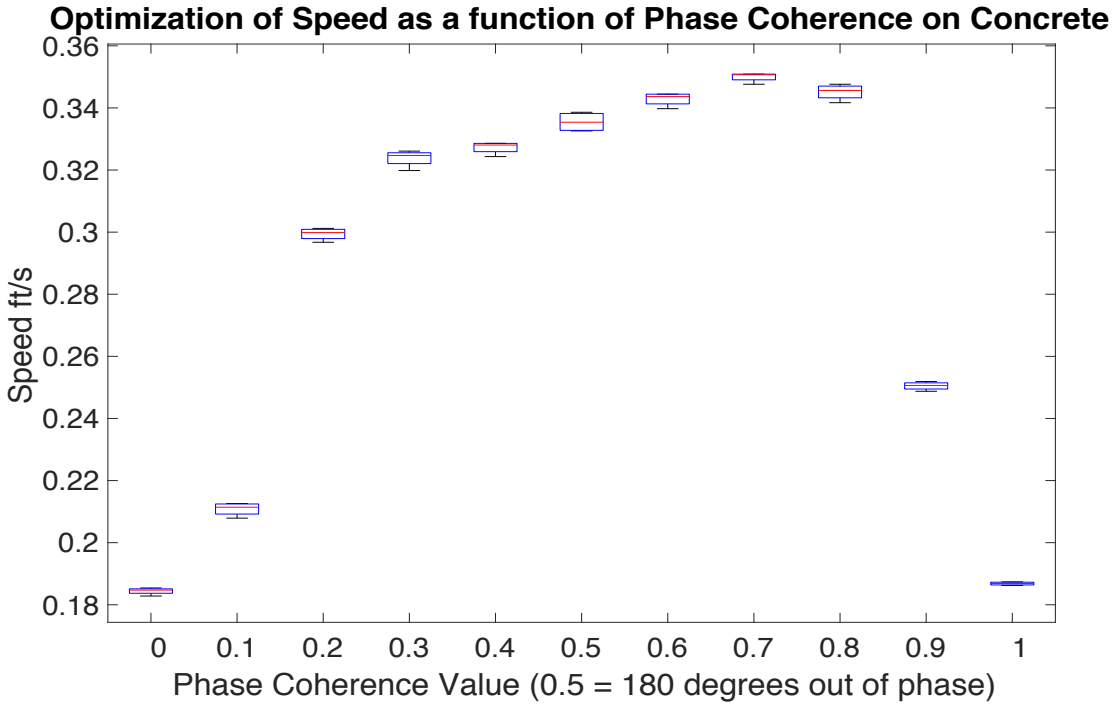

Figure 3.2: Velocity as a function of phase angle difference for a 2 sets of 3 legs walking gait on concrete. This was plotted by the default boxplot in Matlab. The red lines indicate the average, inside the box represents 50% of the data, and then 25% above and below 50%. There are no error bars on phase coherence, the width of the points has no significance.

Dependence of walking speed on walking period. Another input parameter that was tested was the walking period. We swept a range of walking periods between 400 ms and 2000 ms in equal steps of 320 ms, with multiple trials per walking period. This information is displayed in figure 3.3. This figure demonstrates that the maximum velocity is at 720. We also see that there is substantial increase in trial to trial variability in the speed measurements as the walking period drops below

1000 ms. Values tested below about 400 ms, specifically 350 ms, result in immobility of the robot producing no results. This is perhaps expected as when the walking period approaches 0, instantaneous rotation of the leg would be needed, resulting in a singularity; the motor and controller do not have the capability to achieve these quick cycles. Additionally, given the large trial to trial variability for a given set of parameters, traditional optimization algorithms can find this optimization challenging, as they don't know what the true performance is and the slopes and gradients can only be approximately estimated.

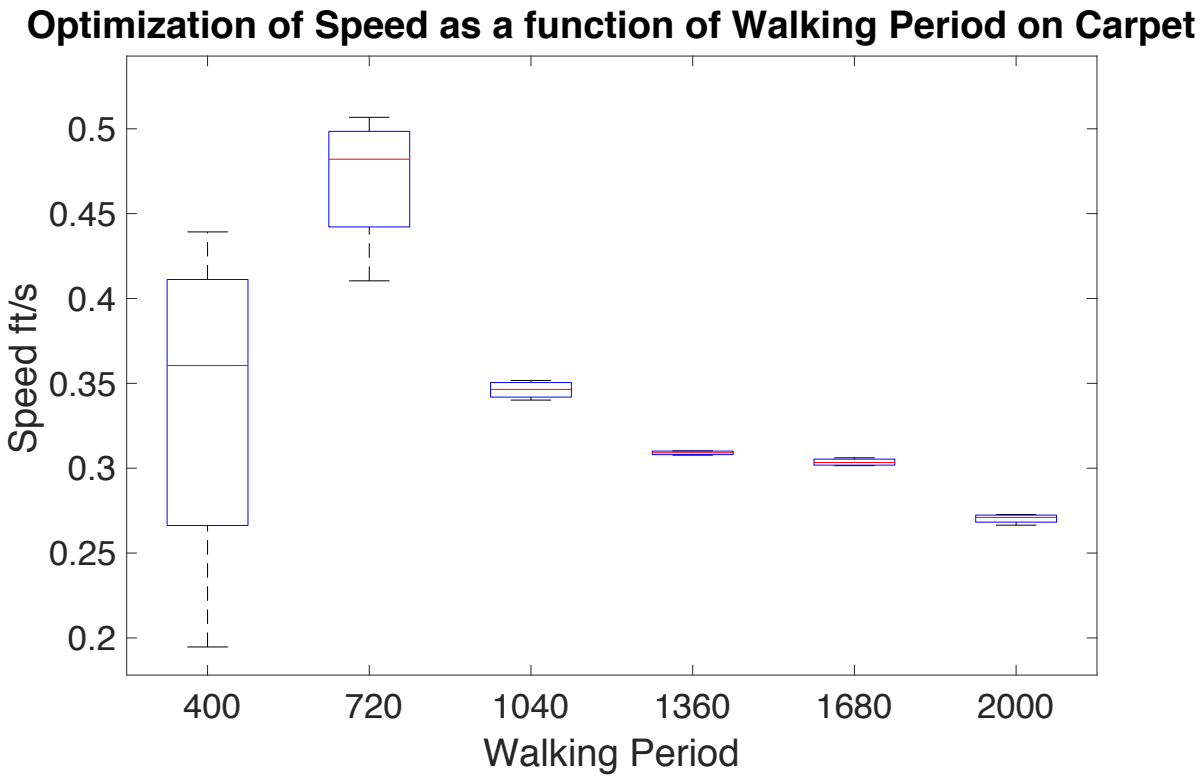

Figure 3.3: Speed as a function of walking period on carpet. Though all of the tests were conducted on carpet, the test point at 400 is on a different carpet surface compared to all of the other data points. This was plotted by the default boxplot in Matlab. The red lines indicate the average, inside the box represents 50% of the data, and then 25% above and below 50%. There are no error bars on phase coherence, the width of the points has no significance.

Dependence of walking speed on PD Gains. Proportional and derivative gains were swept across various values to see the effect of changing them. We did this by systematically changing the P gains and dependently varied the D gain according to the P gain. The derivative gain was dependent as follows: $New\ D\ Gain = Nominal\ D\ Gain * \sqrt{\frac{New\ P\ Gain}{Nominal\ P\ Gain}}$. That is, the derivative gain was changed proportional to the square root of the proportional gain, so that the damping ratio is roughly preserved. The speed is maximized at a P value of 0.025 with a D gain value of 0.91287 as shown in figure 3.7. As expected, the speed drops as the gain value varies from the tuned value. Outside the range in figure 3.7, the tests were inconsistent in repeatability.

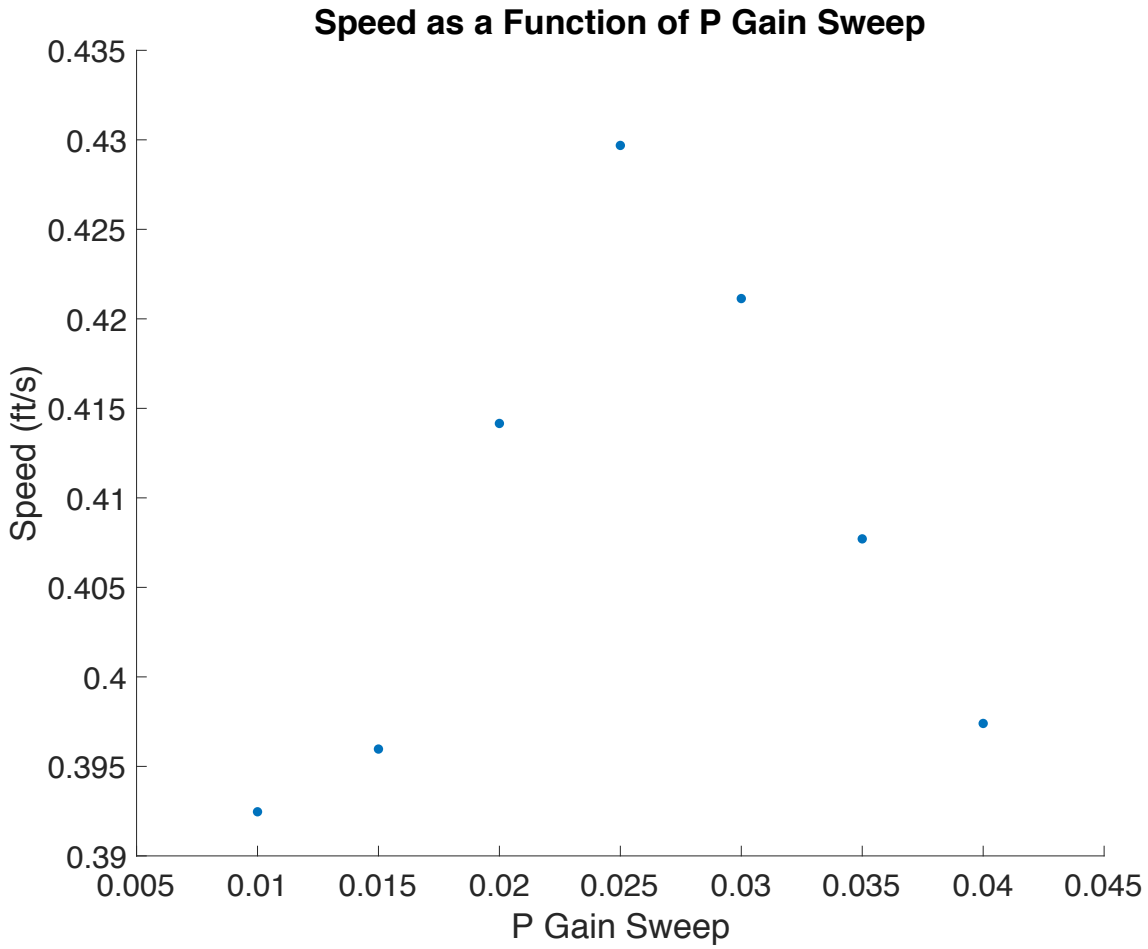

Figure 3.4: This plot shows all nominal values with only P and D gain values changed. The nominal values for the P and D gains were 0.03 and 1, respectively.

Gradient descent in 2D. Given that gradient descent in 1D involves just a subset of the one dimensional parameter sweeps described thus far, here, we describe results from a 2D optimization using gradient descent. Figure 3.4 shows the sequence of iterates from a 2-D optimization. The chosen gait involved the front two legs moving together, the middle two legs moving together and the back two legs moving together. Gradient descent was implemented for changing the phase of the front legs and the phase of the back legs; the phase angle of the middle legs was held constant to achieve a two-input optimization. The initial phases were set to the following: $\emptyset_{\text{front}} = 0.3, \emptyset_{\text{middle}} = 0$, and $\emptyset_{\text{back}} = 0.7$. The front and back phases were changed individually to compute the gradient at that step. Using the gradient of each phase, a new iterate was calculated proportional to the magnitude of the gradients. See figure 3.4. The same process occurs again at this new point to obtain the next iterate. Figure 3.5 shows the sequence of speeds at each iteration and function evaluation for gradients. We measured the time it took for the robot to travel a distance of 55 inches and calculated the speed. The points denoted by 1.1, 2.1, and 3.1 represent the initial guess (1.1) and the next two iterates (2.1 and 3.1). The points 2.1 and 3.1 were calculated by taking gradient descent steps from 1.1 and 2.1, respectively, after finding the gradient at each of those points. The points 1.2 and 1.3 were considered to find the gradient at point 1.1, the points 2.2 and 2.3 were used to compute the gradient at 2.1, and so on. In just 2 iterations from the initial guess location, we were able to increase the speed from 0.2149 ft/s to 0.3499 ft/s. This is a 61.4% increase in speed.

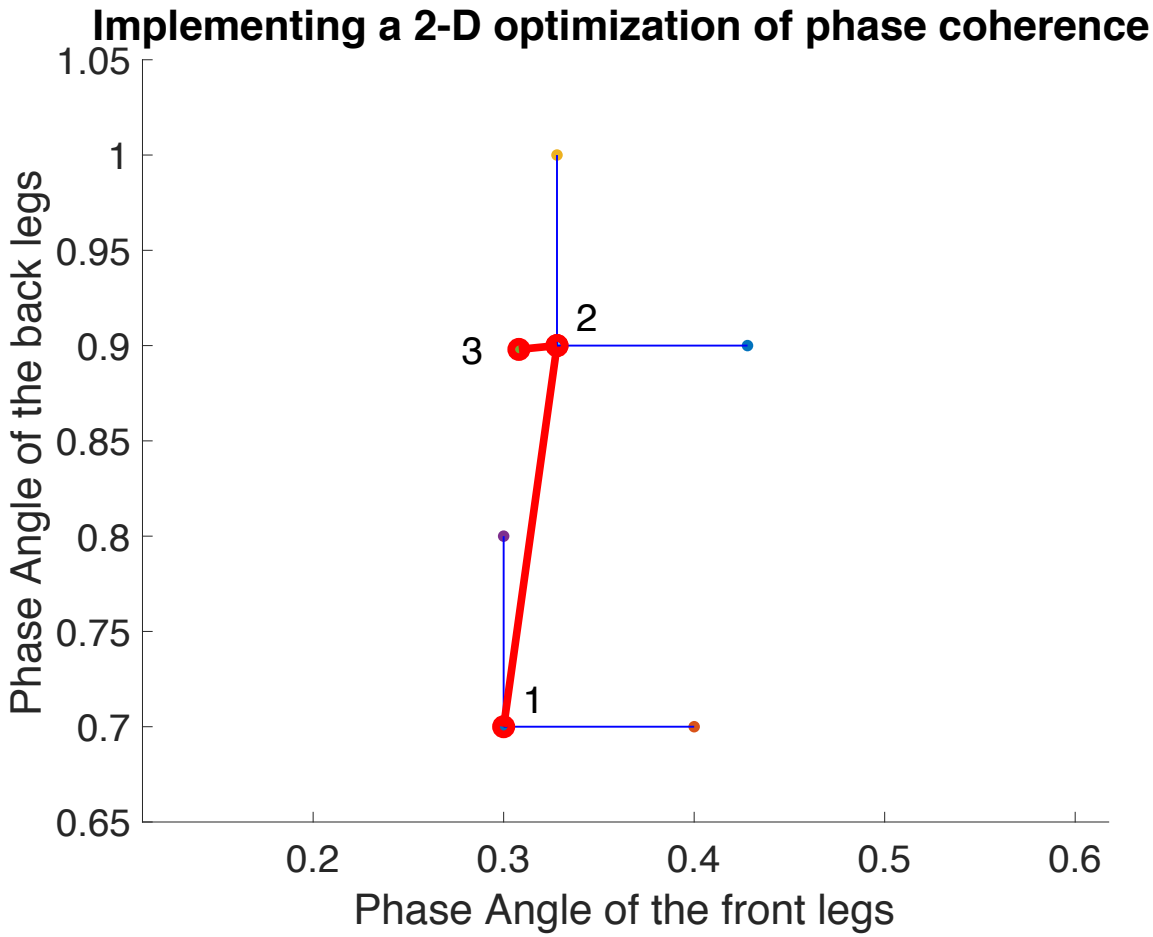

Figure 3.5: This figure shows the path the optimization took based on the gradient calculated at each perturbation. 1 was the initial guess tested. Both parameters, front legs and back legs, were varied independently (blue lines) to calculate the gradient at the initial guess. This gradient was then used to find point 2. This was repeated to calculate the location of point 3.

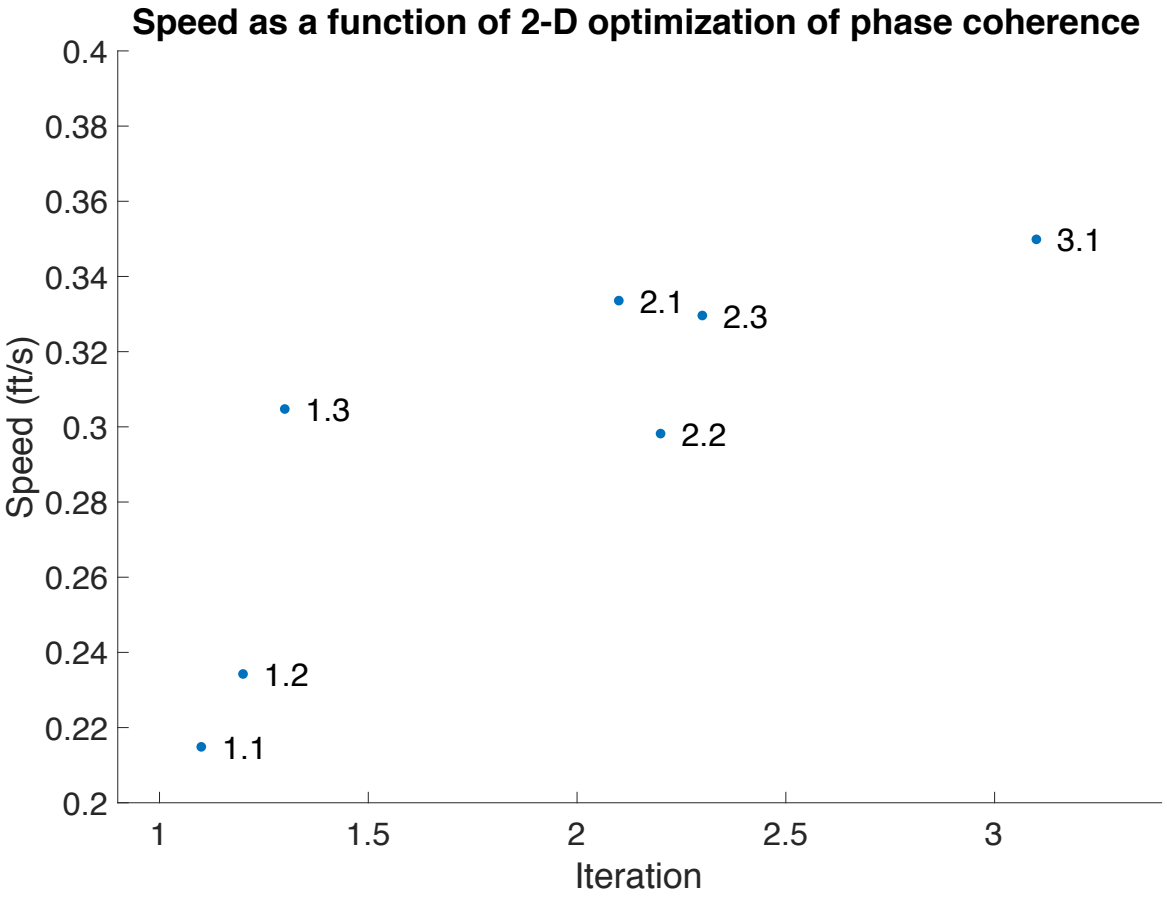

Figure 3.6: Point 1.1 corresponds to point 1, the initial guess, in figure 3.7. Points 1.2 and 1.3 are the independent perturbations tested to calculate the gradient. The gradient at points 1.1 is then used to find point 2.1. This process is then repeated to find point 3.1.

Chapter 4: Discussion & Conclusion

In summary, this project looked at optimization of locomotion of a hexapod robot in various surface conditions. This was accomplished by first constructing MiniRHex, interfacing to its motors through a microcontroller, and then calibrating and initializing the robot. Then, we evaluated the dependence of MiniRHex's walking speed on two control parameters by performing sweeps of one single parameter at a time. This was then repeated on a different surface to compare optimal values of the sweep. We found that the optimal control parameters depended on the surface. Such sweeping through a single parameter is feasible, though time consuming. Real robots have many control parameters, so to optimize MiniRHex on many parameters quickly, an optimization algorithm known as gradient descent was implemented. Though limited testing was conducted using gradient descent, the results were promising in how quickly the solution improved in performances. In just two iterations of the gradient descent method, the speed changed from 0.2149 ft/s to 0.3499 ft/s, a 61.4% increase.

There were challenges and limitations that prevented the project from furthering the data set and perfecting the optimization methodology, to be explored further in future work. Initially, the hardware component of the robot, involving non-standard screws, caused delay in the project --- pointing to the inherent uncertainty and challenges in such build projects, even ones that involve open source information. One limitation of the physical robot and its controller was the open loop direction control of the robot. So the robot did not have directional stability, did not always travel in a straight line, and would sometimes veer off in some direction. The robot did not have position or directional feedback to correct for any path disturbance; one could include sensors to provide

directional control. All path disturbances were largely similar across trials, so that this may not contribute to random errors.

Only low dimensional optimization of input parameters were tested on MiniRHex. In future iterations of this project, higher dimensional optimization should be tested, so as to measure the robustness of the optimization algorithm and its scalability to more parameters. Specifically, we wish to simultaneously vary the relative phases of all the legs so as to discover the optimal relative phases.

More input parameters should be tested in addition to the parameters considered here. Parameters to test include both controller parameters and physical parameters. This list includes, but not limited to: PD gains, phase, walking period, duty cycle, using a PID walking controller rather than a PD controller, weight of robot, leg properties (stiffness, friction, length, and curvature), as well as exploring other metrics of performance such as energy efficiency and reduce variability of measurements.

Finally, we can consider improvements to our implementations of the optimization algorithm. For instance, we used only a small number of gradient descent iterations and did not take the algorithm to convergence. While we obtained dramatic performance improvement with just the few iterations, in future work, we will use more iterations of the optimization algorithm to demonstrate convergence to the true optimality, rather than just shown improvement. Here, we only used gradient descent; other optimization algorithms could be tested to evaluate their relative effectiveness: for instance, Newton's method, CME-ES, conjugate gradient method, etc.

Citations

M. Barragan, N. Flowers, and A. M. Johnson. "MiniRHex: A Small, Open-source, Fully Programmable Walking Hexapod". In *Robotics: Science and Systems Workshop on ``Design and Control of Small Legged Robots"*, Pittsburgh, PA, June 2018.

Bhounsule, P. Cortell, J. Grewal, A. Hendriksen, B. Daniel Karssen, J.G. Paul, C. Ruina, A. "Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge" *The International Journal of Robotics Research.* Vol 33, Issue, 10, 2014.

Collins, S. Ruina, A. "A Bipedal Walking Robot with Efficient and Human-Like Gait." *Proceedings of the 2005 IEEE International Conference on Robotics and Automation,* vol. 2, 2005, pp. 1982-1988.

Estremera, J. Waldron, K. "Thrust Control, Stabilization and Energetics of a Quadruped Running Robot." *The International Journal of Robotics Research*, vol. 27, no. 10, 2008, pp. 1135-1151.

Krishna, P. Prasanth, K. "Energetics of Constant Height Level Bounding in Quadruped Robots." *Robotica*, vol. 34, no. 2, 2016, pp. 403-422.

Ghorbani, R. Wu, Q. "Adjustable Stiffness Artificial Tendons: Conceptual Design and Energetics Study in Bipedal Walking Robots." *Mechanism and Machine Theory*, vol. 44, no. 1, 2009, pp. 140-161.

Fukuoka, Y., Kimura, H., & Cohen, A. H. (2003). Adaptive Dynamic Walking of a Quadruped Robot on Irregular Terrain Based on Biological Concepts. *The International Journal of Robotics Research*, *22*(3–4), 187–202.

Greenblatt, Jeffery B., Saxena, Samveg," Autonomous taxis could greatly reduce greenhouse-gas emissions of US light-duty vehicles" *Nature Climate Change,* vol. 5

Hian Kai Kwa, J. H. Noorden, M. Missel, T. Craig, J. E. Pratt and P. D. Neuhaus, "Development of the IHMC Mobility Assist Exoskeleton," *2009 IEEE International Conference on Robotics and Automation*, Kobe, 2009, pp. 2556-2562.

Hemker, Thomas, Hajime Sakamoto, Maximilian Stelzer, and Oskar von Stryk. "Hardware-in-the-loop optimization of the walking speed of a humanoid robot." In *Proc. of CLAWAR*, p. 20. 2006.

Hemker, T. Stelzer, M. Von Stryke, O. Sacamoto, H. "Efficient Walking Speed Optimization of a Humanoid Robot" *The International Journal of Robotics Research.* Vol 28, Issue 2, 2009.

Lin, Long-Ji, "Self-improving reactive agents based on reinforcement learning, planning and teaching", *Machine Learning,* no. 3, pp. 293-321, May, 1992.

W. D. Smart and L. Pack Kaelbling, "Effective reinforcement learning for mobile robots," *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, Washington, DC, USA, 2002, pp. 3404-3410 vol.4.

Tucker, V. "Energetic Cost of Locomotion in Animals" *Comparative biochemistry and physiology,* vol. 34, no. 1, 1970, pp. 841-846.

Liu, Z. Wang, L. Chen, C.L. "Energy-Efficiency-Based Gait Control System Architecture and Algorithm for Biped Robots" *IEEE transactions on systems, man and cybernetics. Part C, Applications and reviews,* vol. 42, 2012, pp. 926 – 933.

Weingarten et al. "Automated gait adaptation for legged robots." Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on. Vol. 3. IEEE, 2004.

Nocedal, J., Wright, S., "Numerical Optimization" Second Edition, Springer Science + Business Media, 2006.

"The MIT DARPA Robotics Challenge Team." *The MIT DARPA Robotics Challenge Team*, drc.mit.edu/.

Appendix

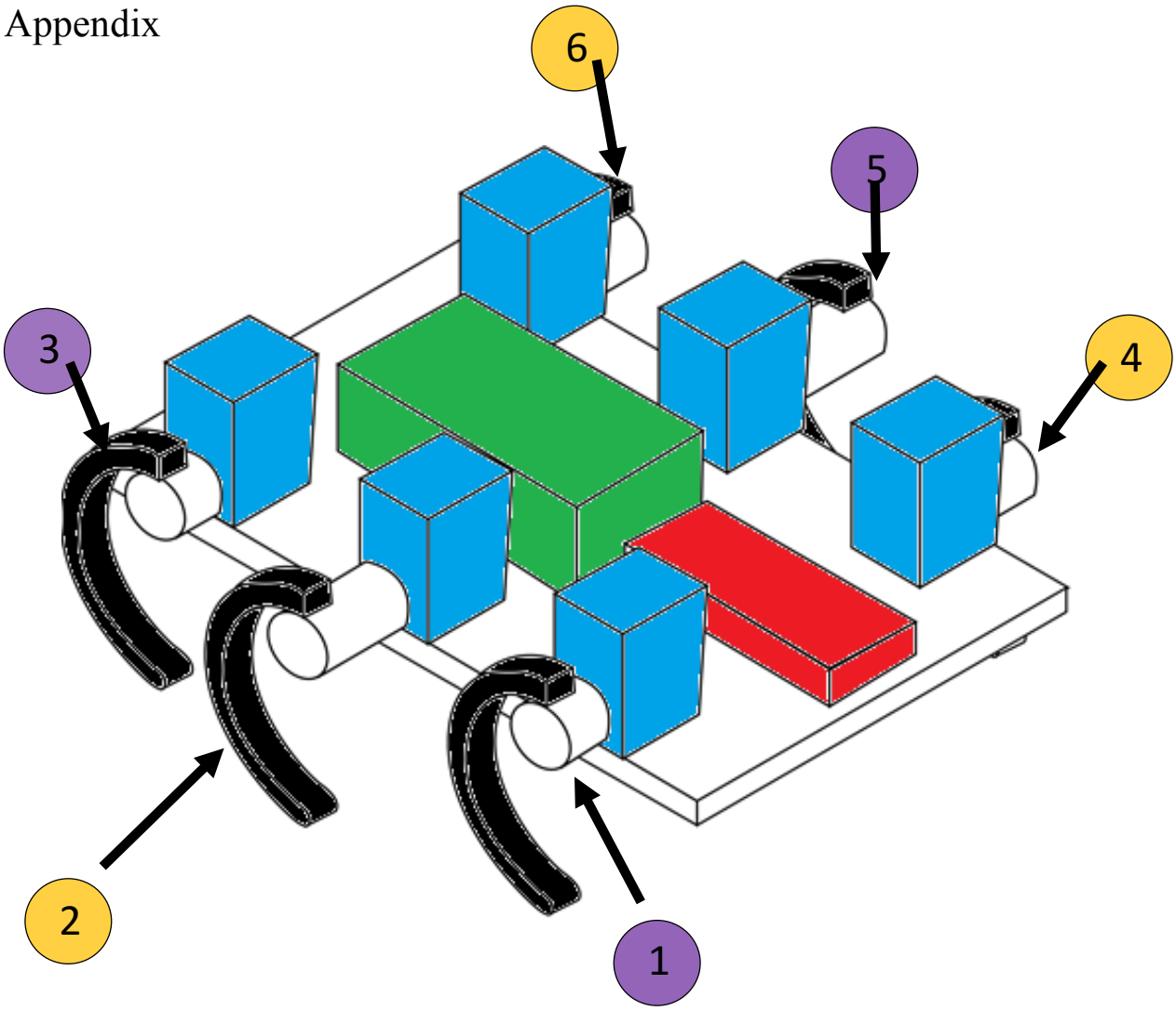

Figure 4.1: MiniRHex leg numbers defined. Figure redrawn from Barragan et al 2018.

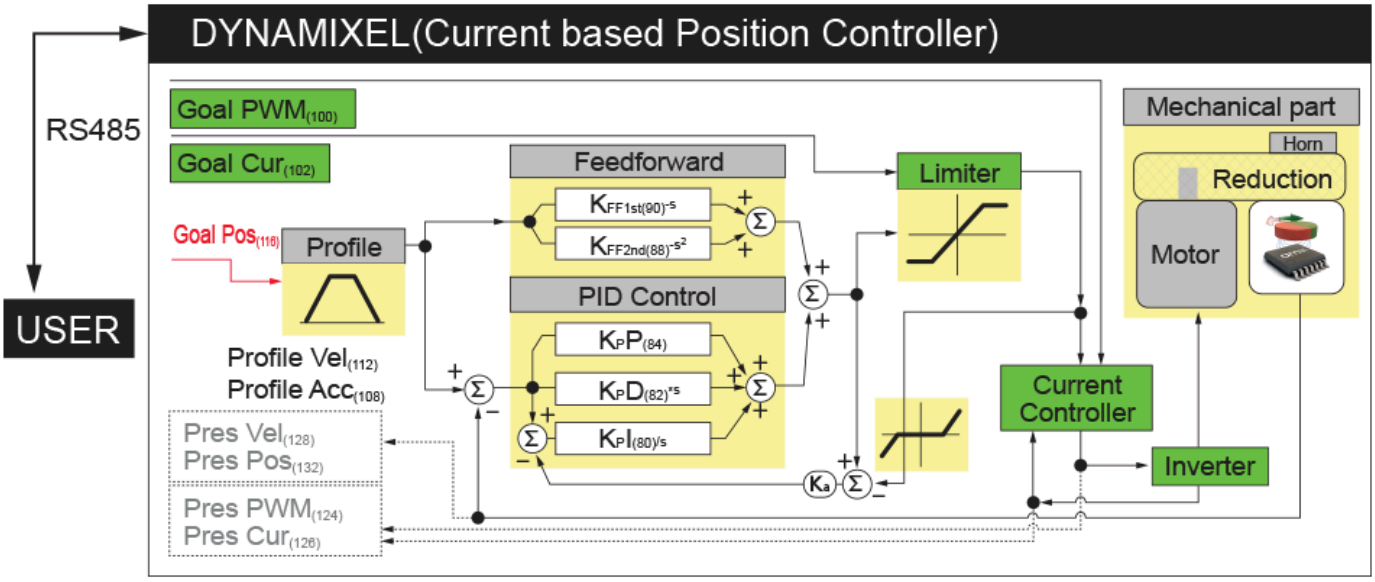

Figure 4.2: Dynamixel XL motor controller block diagram. Figure from Dynamixel user manual.

Figure 4.3: Dynamixel XL-320. Motor that was used in MiniRHex. Used from http://www.robotis.us/dynamixel-xl-320/